Explicit Congestion Notification (ECN) in TCP over Wireless Network

Rohit Ramani *

BAT Group
Silicon Automation Systems Ltd.
Bangalore- 560008, India
rrohit@sasi.com

Abhay Karandikar

Department of Electrical Engineering
Indian Institute of Technology–Bombay
Powai, Mumbai - 400076, India
karandi@ee.iitb.ernet.in

ABSTRACT

*Reliable transport protocols like the Transmission Control Protocol (TCP) are tuned to perform well in traditional wireline networks where packet losses occur mostly because of congestion. However, networks with wireless and lossy links also suffer from significant packet losses due to random losses and handoffs. TCP responds to all the packet losses by invoking congestion control algorithm and this results in degraded end-to-end performance in wireless and lossy links. In this paper we suggest a strategy to determine the cause of packet drops in the wireless network running the TCP protocol. Our method is based on modification of Explicit Congestion Notification (*ECN*) in TCP for wireless networks.*

1. INTRODUCTION

Due to rapid advances in the area of wireless communications and the popularity of the Internet, the provision of packet data services for applications like e–mail, web browsing, mobile computing *etc* over wireless is gaining importance. Most of these applications make use of reliable end–to–end transport level services provided by TCP. TCP is one of the most widely used transport layer protocols in Internet protocol suite [1].

Traditionally, TCP was designed for wireline networks where the channel error rates are very low and congestion is the primary cause of packet loss. TCP reacts to packet losses by dropping its congestion window size before retransmitting packets, initiating congestion control or avoidance mechanisms and backing off its retransmission timer. These measures result in a reduction in the load on intermediate links, thereby controlling the congestion in the network.

Unfortunately, when packets are lost in networks for reasons other than congestion, these measures result in an unnecessary reduction in end–to–end throughput and hence, in suboptimal performance. Communication over wireless links is often characterized by sporadic high bit–error rates and intermittent loss of connectivity due to handoffs. TCP performance in such networks suffers from significant throughput degradation and very high interactive delays.

*This work was performed while the author was at IIT–Bombay

Recently, several schemes have been proposed to alleviate the effects of noncongestion related losses on TCP performances over networks that have wireless or similar high–loss links. These schemes choose from variety of mechanisms, such as local retransmission, split–TCP connection, and forward error correction, to improve end–to–end throughput. However, it is unclear as to what extent each of the mechanisms contributes to the improvement in the performance [2].

There are two different approaches to improve TCP performance in such lossy systems. The first approach hides any noncongestion related losses from the TCP sender, and therefore requires no changes to existing sender implementations. The intuition behind this approach is that, since the problem is local, it should be solved locally, and that the transport layer need not be aware of the characteristics of the individual links. Protocols that adopt this approach attempt to make a lossy link appear as a higher quality link with a reduced effective bandwidth. As a result, most of the losses seen by the sender are caused due to congestion [2]. The second approach attempts to make the sender aware of the existence of the wireless hops, and realizes that some packet losses are not due to congestion. The sender can then avoid invoking congestion control algorithms when noncongestion related losses occur [3]. It is possible for a wireless aware transport protocol to coexist with link layer schemes to achieve good performance.

Our objective is to investigate mechanisms to improve the performance of TCP over wireless networks. The work, reported here, belongs to the second approach discussed above. In the proposed method we are making the sender aware that some of the losses reported are not due to congestion. This is done in our scheme through explicit feedbacks from the network in the form of Explicit Congestion Notification (ECN) about the state of congestion of links in the network. Depending on the feedback received from the network and the packet loss signal from the receiver the sender decides whether the loss is due to congestion or not and then acts accordingly.

In the the next section we propose an extension of Explicit Congestion Notification. In Section 3, we present the results of our simulations. Finally, we conclude this work in Section 4.

2. EXPLICIT CONGESTION NOTIFICATION (ECN) IN TCP FOR WIRELESS LINKS

2.1. TCP's Congestion Control Mechanism

At all times t, the TCP sender maintains the following variables for each connection:

$A(t)$ the lower window edge. All data numbered up to and including $A(t) - 1$ have been transmitted and ACKed. $A(t)$ is nondecreasing and the receipt of an ACK with sequence number n greater than $A(t)$ causes $A(t)$ to jump to n.

$W(t)$ the congestion window (cwnd). The transmitter can send packets with sequence numbers n, $A(t) \leq n \leq A(t) + W(t)$. $W(t) \leq W_{max}$, where W_{max} is the maximum window size advertised by the receiver at the connection time.

W_{th} the slow start threshold (ssthresh).

The TCP receiver is required to generate a duplicate ACK when an out–of–order segment is received. The purpose of this duplicate ACK is to let the TCP sender know that a segment was received out of order and also to inform about the expected sequence number. Since the sender doesn't know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs (for TCP Tahoe it is three) to be received. In TCP Tahoe, if three duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. The sender next sets $W_{th}(t^+)$ to one half of the minimum of the current congestion window $W(t)$ and the receiver's advertised window. Then it retransmits the missing segment. $W(t^+)$ is then set to $W_{th}(t)$ plus 3 times the packet segment size. However if timeout occurs, then $W_{th}(t^+)$ is set to one half of $W(t)$ and $W(t^+)$ is set to one [1].

2.2. Queue Management in Routers

TCP's congestion control algorithm is necessary and powerful but is not enough to provide good service in all circumstances since it treats the network as a black box. Some control mechanism like Active Queue Management is required from the router to complement the end system congestion control mechanism [5].

Drop Tail: Queue management algorithms traditionally manage the length of packet queues in the router by dropping packets only when the buffer overflows. A maximum length for each queue is configured. The router will accept packets till this maximum size is exceeded, at which point it will drop incoming packets. New packets are accepted if the buffer space allows. This technique is known as *Drop Tail*. This method has been used in the Internet for several years [5].

Since all arriving packets from all flows are dropped when the buffer overflows, this interacts badly with the congestion control mechanism of the TCP. A cycle is formed with a burst of packet drops (after the maximum queue size is exceeded) followed by a period of under–utilization at the router as end systems back off. End systems then increase their windows simultaneously up to a point where a burst of packets get dropped again. This phenomenon is called *Global Synchronization*. It leads to poor link utilization and lower overall throughput.

Another problem with *Drop Tail* is that a single connection for a few flows, in some circumstances could monopolize the queue space. This results in a lock out phenomenon leading to synchronization or other timing effects.

Finally, in *Drop Tail*, the queues remain full for long periods or time. One of the major goals of queue management is to reduce the steady state queue size.

Other queue management techniques include *random drop on full* and *drop front on full*.

Random Early Detection (RED) : Active queue management mechanisms detect congestion **before** the queue overflows and provides an indication of this congestion to the end nodes. With this approach TCP does not have to rely on buffer overflow as an indication of congestion since notification happens before serious congestion occurs. One such active management technique is Random Early Detection (RED) [5].

A RED router signals incipient congestion to TCP by dropping packets probabilistically before the queue runs out of buffer space. This drop probability is dependent on a running average queue size to avoid any bias against bursty traffic. A RED router randomly drops arriving packets, with the result that the probability of dropping a packet belonging to a particular flows is approximately proportional to the flow's share of the bandwidth. Thus, if the sender is using relatively more bandwidth, it gets penalized by having more of its packet dropped.

RED operates by maintaining two levels of thresholds minimum (min_{th}) and maximum (max_{th}). It drops packet probabilistically if and only if the average queue size lies between the min_{th} and max_{th} thresholds. If the average queue size is above maximum threshold, the arriving packet is always dropped. When the average queue size is between the minimum and the maximum threshold, each arriving packet is dropped with probability p_a, where p_a is a function of the average queue size. As the average queue length varies between min_{th} and max_{th}, p_a increases linearly towards a configured drop probability max_p. Beyond max_{th}, the drop probability is 100%. Figure 1 illustrates the relationship. Since the dropping is distributed across flows, the problem of *global synchronization* is avoided.

2.3. Explicit Congestion Notification

Explicit Congestion Notification (ECN) is an extension proposed to RED which marks a packet instead of dropping it when the average queue size is between min_{th} and max_{th} [4]. Since ECN marks packets before congestion actually occurs, this is useful for protocols like TCP that are sensitive to even a single packet loss. Upon receipt of a congestion marked packet, the TCP receiver informs the sender (in the subsequent acknowledgement) about incipient congestion which in turn will trigger the congestion avoidance algorithm at the sender. ECN requires support from both the router as well as the end hosts, *i.e.,* the end host TCP stack needs to be modified. If the ECN support is provided then the packets are referred as ECN capable packets. Packets which are not ECN capable will continue to be dropped by RED (as was the case before ECN).

Changes at the Routers: The router side support for ECN can be added by modifying the current RED implementations [6].

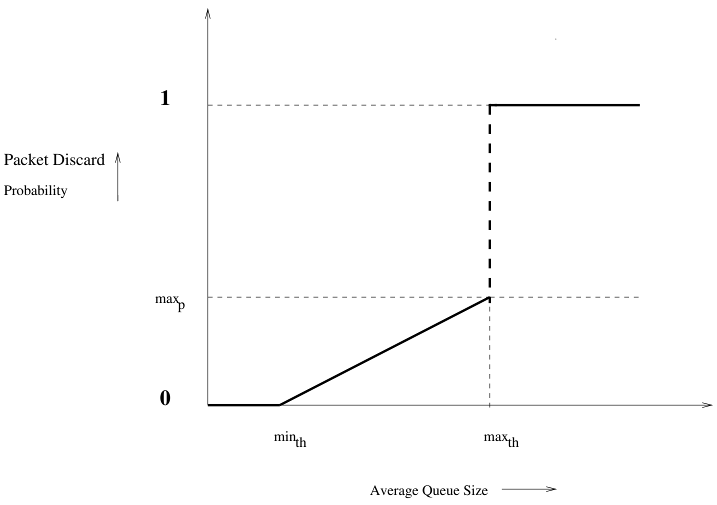

Figure 1: Relationship between the average queue size and the packet dropping probability in a RED router

For ECN capable packets the router marks the packet rather than dropping them (if the average queue size is between min_{th} and max_{th}).

It is necessary that the router identifies that a packet is ECN capable, and should only mark the packets that are ECN capable. This uses two bits in the IP header. The ECN Capable Transport (ECT) bit is set by the sender end systems if both the end systems are ECN capable. In TCP this is confirmed in the pre–negotiation during the connection setup phase (explained in next section). The Congestion Experienced (CE) bit of the packets encountering congestion (if the average queue size is between min_{th} and max_{th}) is marked on their way to the receiver end system with a probability proportional to the average queue size following the procedure used in RED (RFC 2309) routers. Bits 10 and 11 in the IPv6 header are proposed respectively for the ECT and CE bits.

Changes at the TCP Host Side : The proposal to add ECN to TCP specifies two new flags in the reserved field of the TCP header. Bit 9 in the reserved field of the TCP header is designated as the ECN–Echo (ECE) flag which is set by the receiver to indicate the congestion in the network and Bit 8 is designated as the Congestion Window Reduced (CWR) flag which is set by the sender to inform the receiver that it has reacted to its congestion notification.

These two bits are used both for the initialization phase in which the sender and the receiver negotiate the capability and the desire to use ECN, as well as for the subsequent actions to be taken in case there is congestion experienced in the network during the established state.

Following changes need to be made to add ECN to TCP.

1) In the connection setup phase, the source and destination TCPs have to exchange information about their desire and/or capability to use ECN. This is done by setting both the ECE flag and the CWR flag in the SYN packet, which is packet with SYN (synchronize sequence numbers to initiate the connection) flag set, of the initial connection phase by the sender. Upon the receipt of this SYN packet, the receiver will set the ECE flag in the SYN–ACK response. Once this agreement has been reached, the sender will thereafter set the ECT bit in the IP header of data packets for that flow, to indicate to the network that it is capable and willing to participate in ECN. The ECT bit is set on all packets other than pure acknowledgements.

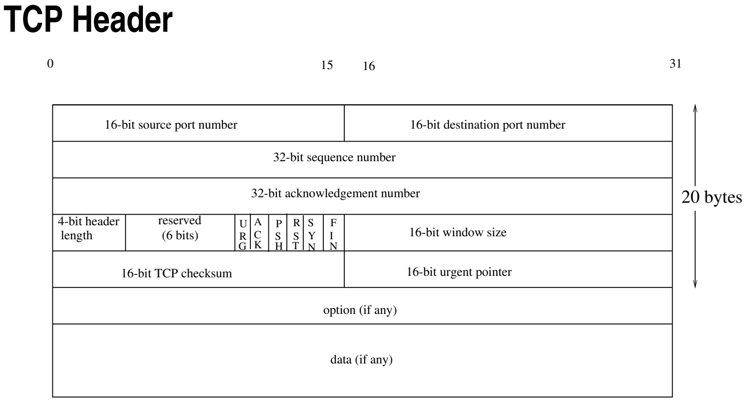

Figure 2: TCP Header

2) When a router decides to drop or mark a packet, it checks the IP–ECT bit in the packet header. It sets the CE bit in the IP header if the IP–ECT bit is set. When such a packet reaches the receiver, the receiver responds by setting the ECE flag (in the TCP header) in the next outgoing acknowledgement for the flow. The receiver will continue to do this in subsequent acknowledgements until it receives from the sender an indication that it (the sender) has responded to the congestion notification.

3) Upon receipt of this ACK, the sender triggers its congestion avoidance algorithm. Once it has taken appropriate steps, the sender sets the CWR bit on the next outgoing packet to inform the receiver that it has reacted to the receiver's notification's of congestion. If there is no new congestion in the network then the receiver stop sending the congestion notifications to the sender.

Note that the sender's reaction to the indication of congestion in the network (when it receives an ACK packet that has the ECE flag set) is equivalent to the Fast Retransmit/Recovery algorithm (when there is a congestion loss) in NON–ECN–capable TCP *ie,* the sender halves the congestion window and reduces the slow start threshold. Fast Retransmit/Recovery is still available for ECN capable TCP for responding to three duplicate acknowledgements.

Changes at the Receiver Side : When TCP receives a CE data packet at the destination end–system, the TCP receiver sets the ECN–Echo flag in the TCP header of the subsequent ACK packet. If there is any ACK withholding implemented, as in current "delayed–ACK" TCP implementations where the TCP receiver can send an ACK for two arriving data packets, then the ECE flag and the CE bit in the ACK packet will be set to the OR of the CE bits of all of the data packets being acknowledged. That is, if any of the received data packet is CE packet, then the returning ACK has the ECE flag and the CE bit set. To provide robustness against the possibility of a dropped ACK packet carrying an ECE flag, the TCP receiver must set the ECE flag in a series of ACK packets. The TCP receiver uses the CWR flag to determine when to stop setting the ECE flag. When an ECN capable TCP reduces its congestion window for any reason (because of a retransmit timeout, a Fast Retransmit, or in response to an ECN), the TCP sets the CWR flag in the TCP header of the first data packet sent after the window reduction. If that data packet is dropped in the network, then the sending TCP will have to reduce the congestion window again and retransmit the dropped packet. Thus, the congestion window reduced message is reliably delivered to the data receiver. After the receipt of the packet with CWR flag set, acknowledgements for subsequent non–CE data packets do not have the ECE flag set. If another CE packet is received by the receiver, the receiver would once again send ACK packets with the ECE flag set. While the receipt of a CWR packet does not guarantee that the sender received the ECE message, this does indicate that the sender reduced its congestion window at some point *after* it sent the data packet for which the CE bit was set.

2.4. Extension of ECN to TCP Over Wireless

TCP interprets every packet loss as caused due to congestion which is not the case in a wireless network with a lossy link.

We propose an extension to ECN to determine the cause of packet drops in a wireless network running the TCP protocol. This is done by observing the Congestion Experienced (CE) bit in the duplicate ACKs received if the TCP is ECN–capable.

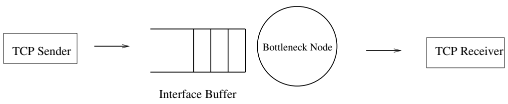

Figure 3: Two Node Wireless Network

To determine the cause of packet drops (Refer Figure 3), we specifically look at

- CE bit in the ACK received.
- Number of duplicate ACKs received.

Consider that the interface buffer of the bottleneck node in Figure 3 is about to overflow. It means its queue length has exceeded the minimum threshold value of the average queue length. If this occurs and the network is ECN–capable, it will set the CE bit in the TCP header of the packet with some probability as discussed earlier. The receiver also sends ACK to the sender with its CE bit set. If the sender receives third duplicate ACK with its CE bit set then the sender can infer that a packet loss has occurred due to congestion in the network and it can start its congestion avoidance algorithm to take care of it. This is because the receiver has already warned the sender by setting the CE bit in the ACK which indicates that the interface buffer at the gateway is about to overflow. When the buffer actually overflows and a packet is dropped then the receiver sends duplicate ACK and the sender can conclude that there is congestion in the network.

Suppose the queue length of the buffer is below the minimum threshold value, then the CE bit in the TCP header will not be set by the bottleneck node and as a result the ACKs coming from the receiver will also not have their CE bit set. Now if there is a random packet loss in the network which is informed to the sender by sending three duplicate ACKs by the receiver in which none of the ACKs have their CE bit set, then the sender can conclude that packet loss is due to random wireless loss in the network. As a result once a loss has been identified as a random loss, congestion avoidance algorithm is not used. However the window size of the sender may still be reduced as as the system is still in the bad state. Hence if the packet loss is detected due to random wireless loss, then the approach is to reduce the window size but recover fast when the connection becomes good again. We have incorporated this approach.

Changes in the Network At the router and the receiver side no changes have to be made. But at the the host side the following changes have to be made. Upon receipt of the ACK, if the CE bit is set and the ACK is the third duplicate ACK in a row then the sender triggers its congestion avoidance algorithm by halving its congestion window cwnd and updating its congestion window threshold value ssthresh. On the other hand if the CE bit is not set and the ACK is the third duplicate acknowledgement then the window size is reduced, but is recovered very fast. Once it has taken these appropriate steps, the sender sets the CWR bit on the next data packet to inform the receiver that it has reacted to the receiver's notification of congestion.

3. SIMULATION RESULTS

The simulation of the proposed algorithm has been performed on NS. The LBNL network simulator, NS, is a simulation tool developed by the Network Research Group at the Lawrence Berkeley National Laboratory. NS is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing, and multicast protocols. We have simulated the wireless topology shown in Figure 3 with the modified TCP implemented.

The capacity of the interface buffer at the bottleneck node is 120kbits. Data rate of the link is 2Mbps and propagation delay is 2 μs. We have used 914MHz Lucent WaveLAN radio interface with the following parameters:

- Power of transmission: 0.2818 W
- Frequency : 914 MHz
- Receive threshold, *ie*, minimum power required to receive a packet: 3.652×10^{-10} W

For RED queue we set min_{th}=96kbits, max_{th}=120kbits, mean packet size= 8kbits and max_p=0.02. The simulation were performed for 60 seconds.

In Figure 4 we have plotted the variation of congestion window size of the sender for both TCP (with drop tail buffer) and modified TCP. Here cwnd1 is for TCP and cwnd2 is for modified TCP.

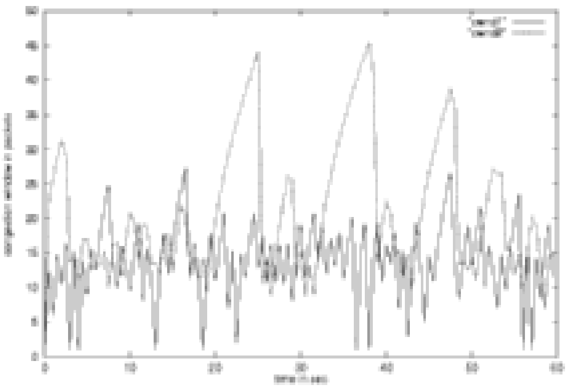

Figure 4: TCP Window Variation

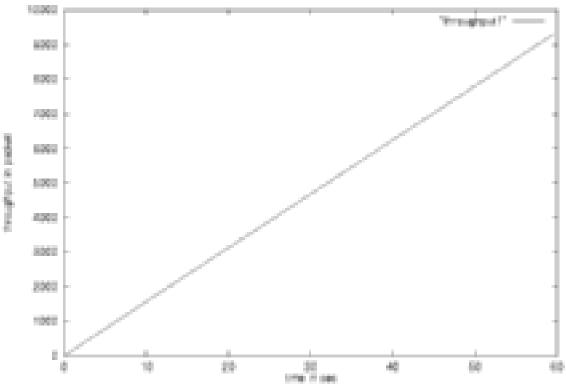

Figure 5: Throughput of unmodified TCP

We see in Figure 4 that the congestion window size for modified TCP is larger than that of unmodified TCP for most of the time.

The throughput versus time for the unmodified TCP and modified TCP is shown in Figure 5 and 6 respectively. In case of modified TCP 30 more packets were sent than unmodified TCP during simulation time.

4. CONCLUSION

In this paper we have suggested modifications to Explicit Congestion Notification (ECN) in TCP to improve the performance of TCP over wireless. We observed significant improvement in the performance of network in our proposed method. There are many parameters in the suggested method which may affect the performance significantly for example min_{th}, max_{th}, max_p *etc* and need to be explored further. One thing which has to be carefully looked in detail is how the window size of the sender should change when it has detected non-congestion related losses. The proposed method has been developed for TCP Tahoe and its possible extensions to various other TCP protocols needs to be studied in detail.

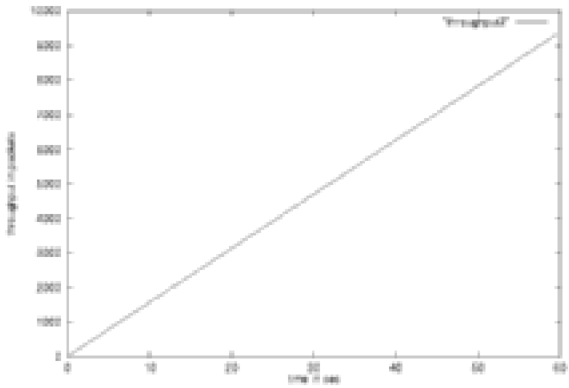

Figure 6: Throughput of modified TCP

5. REFERENCES

[1] W.R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison Wesley, 1994

[2] H. Balakrishnan, V.N. Padamanabhan, S. Sheshan, and R.H. Katz, “A Comparison of Mechanism for Improving TCP Performance over Wireless Links”, *ACM/IEEE Transaction on Networking*, Vol.5, No.6, pp756-769,Dec. 1997.

[3] D. Bansal, A. Chandra, R. Shorey, “An Extension of the TCP Flow Control Algorithm for Wireless Networks”, *ICPWC*, pp207-210,1999.

[4] K. K. Ramakrishna, S. Floyd, “A proposal to add Explicit Congestion Notification (ECN) to IP”, *RFC*2481, January 1999.

[5] S. Floyd, V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance” *IEEE Transaction on Networking*, Vol. 1, No. 4, pp397-413, August 1993,

[6] U. Ahmed, J. H. Salim, “Performance Evaluation of Explicit Congestion (ECN) in IP Networks”, *Internet–Draft*, March 2000, *http://search.ietf.org/internet-drafts/draft-hadi-jhsua-ecnperf-01.txt*

[7] T.V.Lakshman, U. Madhow, “Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss”, *IEEE/ACM Transaction on Networking*, Vol. 5, No. 3, pp336-35, 1997

[8] Anurag Kumar,“Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link” *IEEE/ACM Transaction on Networking*, Vol. 6, No. 4, pp485-498, August 1998

[9] “Network Simulator”, *http://www-mash.cs.berkeley.edu/ns*